**What is data type :** A type defines a set of values and set of operations that can be applied on those values. The set of values for each type is known as the domain for the type.

For example, a light switch can be compared to a computer type. It has a set of two values**: on** and **off**. Since the domain of a light switch consists of only these two values, its size is two. There are only two operations that can be applied to a light switch, **turn on** and **turn off**.

**What is variable:** A variable is a location in computer memory where a value can be stored for use by a program.

Variable name and value is akin to mailbox name/address and mail/data . respectively.

**Variables are access point to your computer’s memory.**

**Declaration statement:** A declation statement for variable has the general form:

**data\_type variable\_name;**

**int num1;**

**Multiple declaration:** The common form of such a declaration is:

**Data\_type variable\_list;**

int num1; int num2;

Can be replaced by a single declaration

**int num1, num2;**

**Identifiers:** Identifiers are used to give name methods, variable.

Identifiers can be made up of any combination of letters, digits, or underscores ( \_ ) selecting according to the following rules:

1, The first character of an identifier must be a letter or underscore (\_).

2. Only letters, digits, or underscores may follow the initial letter. Blank spaces are not allowed; use underscores to separate words in an identifier consisting of multiple words.

3. An identifier cannot be one of the keywords.

**Assignment statements: this statement has the general form:**

**variable = operand;** ( = Assignment operator (AO) )

The assignment operator (AO) in an assignment statement tells the computer

1. First to determine the value of the operand to the right of the AO and
2. then to store( or assign) that value in the location associated with the variable to the left of the AO

Examples of assignments using expressions containing these operators are:

**Sum = 3 +7; slope = (y2- y1) / ( x2 –x1 ); product = 0.05 \* 14.6 ;**

**Variables and Declarations:**

**Put 62 in Location 1600**

**Put 17 in Location 1604**

**num1 num2 sum**

79

17

62

**6262**

**1600 1604 1608 => Memory Location**

**Arithmetic Operators: The operators used for arithmetic operations are called arithmetic operators, and listed below: in Table 1.**

**Table 1.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Java Operation** | **Arithmetic operation** | **Algebraic expression** | **Java expression** |
| **Addition** | **+** | **F+7** | **F+7** |
| **Subtraction** | **-** | **p-c** | **p-c** |
| **Multiplication** | **\*** | **bm** | **b \*m** |
| **Division** | **/** | **or x ÷y** | **x/y** |
| **Modulus** | **%** | **r mod s** | **r % s** |

**The division operator ( /) computes the integer quotient of its two operands. Some examples,**

**6/3 =2**

**7/3= 2 3 “ goes into” 7 twice**

**8/3=2 , the quotient is not rounded**

**3/4= 0**

**3/0 = is an error**

**Precedence of arithmetic operators:**

**Evaluate the expression:**

1. **y = mx + c**
2. **z= pr % q + w/r –y**
3. **y = ax⁴ +bx +c**

**Schematic representation of the action of the mod operator:**

**The mod operator, applied to two positive integers, computes the**

**Remainder when the first is divided by the second.**

**0 mod 8= 0**

**1 mod 8= 1**

**2 mod 8= 2**

**3 mod 8= 3**

**4 mod 8= 4**

**5 mod 8= 5**

**6 mod 8= 6**

**7 mod 8= 7**

**8 mod 8= 0**

**9 mod 8= 1**

**10 mod 8= 2**

**11 mod 8= 3**

**12 mod 8= 4**

**13 mod 8= 5**

**14 mod 8= 6**

**15 mod 8= 7**

**The result produced by the mod operator is a number in the range**

**0<= ( a mod b) < b.**

**.**

**.**

**.**

**Assignment variations: Although only one variable is allowed immediately to the left of the equal sign (AO) in an assignment expression, the variable on the left of the equal sign ( AO) can also be used on the right of the equal sign.**

**For example, sum = sum +10; is not an Equation.**

**Assignment expressions like sum= sum + 10; sum = sum -10; sum = sum \* 10;**

**Sum = sum /10; sum = sum % 10; which use the same variable on both sides of the assignments operator, can be written using the following shortcut assignment operators: += -= \*= /= %=**

**sum +=10; sum -=10; sum \*=10; sum /=10; sum %=10;**